mistune Documentation

Release 0.8.4

Hsiaoming Yang

Dec 06, 2018

Contents

1 Features **3**

2 Installation **5**

3 Basic Usage **7**

4 Options **9**

5 Renderer **11**
- 5.1 Block Level . . . 11
- 5.2 Span Level . . . 12
- 5.3 Footnotes . . . 12

6 Lexers **13**

7 Contribution & Extensions **15**

8 Developer Guide **17**

9 Changelog **21**
- 9.1 Version 0.8.4 . . . 21
- 9.2 Version 0.8.3 . . . 21
- 9.3 Version 0.8.2 . . . 21
- 9.4 Version 0.8.1 . . . 22
- 9.5 Version 0.8 . . . 22
- 9.6 Version 0.7.4 . . . 22
- 9.7 Version 0.7.3 . . . 22
- 9.8 Version 0.7.2 . . . 22
- 9.9 Version 0.7.1 . . . 23
- 9.10 Version 0.7 . . . 23
- 9.11 Version 0.6 . . . 23
- 9.12 Version 0.5.1 . . . 23
- 9.13 Version 0.5 . . . 23
- 9.14 Version 0.4.1 . . . 24
- 9.15 Version 0.4 . . . 24
- 9.16 Version 0.3.1 . . . 24
- 9.17 Version 0.3 . . . 24
- 9.18 Version 0.2 . . . 24

9.19 Version 0.1 24

Python Module Index **25**

The fastest markdown parser in pure Python with renderer features, inspired by marked.

CHAPTER 1

Features

- **Pure Python**. Tested in Python 2.7, Python 3.5+ and PyPy.
- **Very Fast**. It is the fastest in all **pure Python** markdown parsers.
- **More Features**. Table, footnotes, autolink, fenced code etc.

View the benchmark results.

CHAPTER 2

Installation

Installing mistune with pip:

```
$ pip install mistune
```

Mistune can be faster, if you compile with cython:

```
$ pip install cython mistune
```

CHAPTER 3

Basic Usage

A simple API that render a markdown formatted text:

```
import mistune

mistune.markdown('I am using **mistune markdown parser**')
# output: <p>I am using <strong>mistune markdown parser</strong></p>
```

If you care about performance, it is better to re-use the Markdown instance:

```
import mistune

markdown = mistune.Markdown()
markdown('I am using **mistune markdown parser**')
```

Mistune has enabled all features by default. You don't have to configure anything. But there are options for you to change the parser behaviors.

CHAPTER 4

Options

Here is a list of all options that will affect the rendering results, configure them with `mistune.Renderer`:

```
renderer = mistune.Renderer(escape=True, hard_wrap=True)
# use this renderer instance
markdown = mistune.Markdown(renderer=renderer)
markdown(text)
```

- **escape**: if set to *False*, all raw html tags will not be escaped.
- **hard_wrap**: if set to *True*, it will has GFM line breaks feature. All new lines will be replaced with `<br>` tag
- **use_xhtml**: if set to *True*, all tags will be in xhtml, for example: `<hr />`.
- **parse_block_html**: parse text only in block level html.
- **parse_inline_html**: parse text only in inline level html.

When using the default renderer, you can use one of the following shortcuts:

```
mistune.markdown(text, escape=True, hard_wrap=True)

markdown = mistune.Markdown(escape=True, hard_wrap=True)
markdown(text)
```

CHAPTER 5

Renderer

Like misaka/sundown, you can influence the rendering by custom renderers. All you need to do is subclassing a *Renderer* class.

Here is an example of code highlighting:

```
import mistune
from pygments import highlight
from pygments.lexers import get_lexer_by_name
from pygments.formatters import html

class HighlightRenderer(mistune.Renderer):
    def block_code(self, code, lang):
        if not lang:
            return '\n<pre><code>%s</code></pre>\n' % \
                mistune.escape(code)
        lexer = get_lexer_by_name(lang, stripall=True)
        formatter = html.HtmlFormatter()
        return highlight(code, lexer, formatter)

renderer = HighlightRenderer()
markdown = mistune.Markdown(renderer=renderer)
print(markdown('```python\nassert 1 == 1\n```'))
```

Find more renderers in mistune-contrib.

5.1 Block Level

Here is a list of block level renderer API:

```
block_code(code, language=None)
block_quote(text)
block_html(html)
```

(continues on next page)

(continued from previous page)

```
header(text, level, raw=None)
hrule()
list(body, ordered=True)
list_item(text)
paragraph(text)
table(header, body)
table_row(content)
table_cell(content, **flags)
```

The *flags* tells you whether it is header with `flags['header']`. And it also tells you the align with `flags['align']`.

5.2 Span Level

Here is a list of span level renderer API:

```
autolink(link, is_email=False)
codespan(text)
double_emphasis(text)
emphasis(text)
image(src, title, alt_text)
linebreak()
newline()
link(link, title, content)
strikethrough(text)
text(text)
inline_html(text)
```

5.3 Footnotes

Here is a list of renderers related to footnotes:

```
footnote_ref(key, index)
footnote_item(key, text)
footnotes(text)
```

CHAPTER 6

Lexers

Sometimes you want to add your own rules to Markdown, such as GitHub Wiki links. You can't achieve this goal with renderers. You will need to deal with the lexers, it would be a little difficult for the first time.

We will take an example for GitHub Wiki links: `[[Page 2|Page 2]]`. It is an inline grammar, which requires custom `InlineGrammar` and `InlineLexer`:

```
import copy,re
from mistune import Renderer, InlineGrammar, InlineLexer

class WikiLinkRenderer(Renderer):
    def wiki_link(self, alt, link):
        return '<a href="%s">%s</a>' % (link, alt)

class WikiLinkInlineLexer(InlineLexer):
    def enable_wiki_link(self):
        # add wiki_link rules
        self.rules.wiki_link = re.compile(
            r'\[\['                     # [[
            r'([\s\S]+?\|[\s\S]+?)'     # Page 2|Page 2
            r'\]\](?!\])'               # ]]
        )

        # Add wiki_link parser to default rules
        # you can insert it some place you like
        # but place matters, maybe 3 is not good
        self.default_rules.insert(3, 'wiki_link')

    def output_wiki_link(self, m):
        text = m.group(1)
        alt, link = text.split('|')
        # you can create an custom render
        # you can also return the html if you like
        return self.renderer.wiki_link(alt, link)
```

You should pass the inline lexer to `Markdown` parser:

```
renderer = WikiLinkRenderer()
inline = WikiLinkInlineLexer(renderer)
# enable the feature
inline.enable_wiki_link()
markdown = Markdown(renderer, inline=inline)
markdown('[[Link Text|Wiki Link]]')
```

It is the same with block level lexer. It would take a while to understand the whole mechanism. But you won't do the trick a lot.

CHAPTER 7

Contribution & Extensions

Mistune itself doesn't accept any extension. It will always be a simple one file script.

If you want to add features, you can head over to mistune-contrib.

Here are some extensions already in mistune-contrib:

- Math/MathJax features
- Highlight Code Renderer
- TOC table of content features
- MultiMarkdown Metadata parser

Get inspired with the contrib repository.

CHAPTER 8

Developer Guide

Here is the API reference for mistune.

class `mistune.`**`Renderer`**(***kwargs*)

The default HTML renderer for rendering Markdown.

autolink(*link*, *is_email=False*)

Rendering a given link or email address.

Parameters

- **link** – link content or email address.
- **is_email** – whether this is an email or not.

block_code(*code*, *lang=None*)

Rendering block level code. `pre > code`.

Parameters

- **code** – text content of the code block.
- **lang** – language of the given code.

block_html(*html*)

Rendering block level pure html content.

Parameters **html** – text content of the html snippet.

block_quote(*text*)

Rendering <blockquote> with the given text.

Parameters **text** – text content of the blockquote.

codespan(*text*)

Rendering inline *code* text.

Parameters **text** – text content for inline code.

double_emphasis(*text*)

Rendering **strong** text.

Parameters **text** – text content for emphasis.

emphasis(*text*)
Rendering *emphasis* text.

Parameters **text** – text content for emphasis.

escape(*text*)
Rendering escape sequence.

Parameters **text** – text content.

footnote_item(*key*, *text*)
Rendering a footnote item.

Parameters

- **key** – identity key for the footnote.
- **text** – text content of the footnote.

footnote_ref(*key*, *index*)
Rendering the ref anchor of a footnote.

Parameters

- **key** – identity key for the footnote.
- **index** – the index count of current footnote.

footnotes(*text*)
Wrapper for all footnotes.

Parameters **text** – contents of all footnotes.

header(*text*, *level*, *raw=None*)
Rendering header/heading tags like `<h1>` `<h2>`.

Parameters

- **text** – rendered text content for the header.
- **level** – a number for the header level, for example: 1.
- **raw** – raw text content of the header.

hrule()
Rendering method for `<hr>` tag.

image(*src*, *title*, *text*)
Rendering a image with title and text.

Parameters

- **src** – source link of the image.
- **title** – title text of the image.
- **text** – alt text of the image.

inline_html(*html*)
Rendering span level pure html content.

Parameters **html** – text content of the html snippet.

linebreak()
Rendering line break like `<br>`.

link (*link*, *title*, *text*)
Rendering a given link with content and title.

Parameters

- **link** – href link for `<a>` tag.
- **title** – title content for *title* attribute.
- **text** – text content for description.

list (*body*, *ordered=True*)
Rendering list tags like `<ul>` and `<ol>`.

Parameters

- **body** – body contents of the list.
- **ordered** – whether this list is ordered or not.

list_item (*text*)
Rendering list item snippet. Like `<li>`.

newline ()
Rendering newline element.

paragraph (*text*)
Rendering paragraph tags. Like `<p>`.

placeholder ()
Returns the default, empty output value for the renderer.

All renderer methods use the '+=' operator to append to this value. Default is a string so rendering HTML can build up a result string with the rendered Markdown.

Can be overridden by Renderer subclasses to be types like an empty list, allowing the renderer to create a tree-like structure to represent the document (which can then be reprocessed later into a separate format like docx or pdf).

strikethrough (*text*)
Rendering ~~strikethrough~~ text.

Parameters **text** – text content for strikethrough.

table (*header*, *body*)
Rendering table element. Wrap header and body in it.

Parameters

- **header** – header part of the table.
- **body** – body part of the table.

table_cell (*content*, ***flags*)
Rendering a table cell. Like `<th>` `<td>`.

Parameters

- **content** – content of current table cell.
- **header** – whether this is header or not.
- **align** – align of current table cell.

table_row (*content*)
Rendering a table row. Like `<tr>`.

Parameters **content** – content of current table row.

text(*text*)
Rendering unformatted text.

Parameters **text** – text content.

class mistune.**Markdown**(*renderer=None*, *inline=None*, *block=None*, ***kwargs*)
The Markdown parser.

Parameters

- **renderer** – An instance of `Renderer`.
- **inline** – An inline lexer class or instance.
- **block** – A block lexer class or instance.

render(*text*)
Render the Markdown text.

Parameters **text** – markdown formatted text content.

mistune.**markdown**(*text*, *escape=True*, ***kwargs*)
Render markdown formatted text to html.

Parameters

- **text** – markdown formatted text content.
- **escape** – if set to False, all html tags will not be escaped.
- **use_xhtml** – output with xhtml tags.
- **hard_wrap** – if set to True, it will use the GFM line breaks feature.
- **parse_block_html** – parse text only in block level html.
- **parse_inline_html** – parse text only in inline level html.

mistune.**escape**(*text*, *quote=False*, *smart_amp=True*)
Replace special characters "&", "<" and ">" to HTML-safe sequences.

The original cgi.escape will always escape "&", but you can control this one for a smart escape amp.

Parameters

- **quote** – if set to True, " and ' will be escaped.
- **smart_amp** – if set to False, & will always be escaped.

CHAPTER 9

Changelog

Here is the full history of mistune.

9.1 Version 0.8.4

Released on Oct. 11, 2018

- Support an escaped pipe char in a table cell. #150
- Fix ordered and unordered list. #152
- Fix spaces between = in HTML tags
- Add max_recursive_depth for list and blockquote.
- Fix fences code block.

9.2 Version 0.8.3

Relased on Dec. 04, 2017

- Fix nested html issue. #137

9.3 Version 0.8.2

Relased on Dec. 04, 2017

- Fix `_keyify` with lower case.

9.4 Version 0.8.1

Released on Nov. 07, 2017

- Security fix CVE-2017-16876, thanks Dawid Czarnecki

9.5 Version 0.8

Released on Oct. 26, 2017

- Remove non breaking spaces preprocessing
- Remove rev and rel attribute for footnotes
- Fix bypassing XSS vulnerability by junorouse

This version is strongly recommended, since it fixed a security issue.

9.6 Version 0.7.4

Released on Mar. 14, 2017

- Fix escape_link method by Marcos Ojeda
- Handle block HTML with no content by David Baumgold
- Use expandtabs for tab
- Fix escape option for text renderer
- Fix HTML attribute regex pattern

9.7 Version 0.7.3

Released on Jun. 28, 2016

- Fix strikethrough regex
- Fix HTML attribute regex
- Fix close tag regex

9.8 Version 0.7.2

Released on Feb. 26, 2016

- Fix *hard_wrap* options on renderer.
- Fix emphasis regex pattern
- Fix base64 image link #80.
- Fix link security per #87.

9.9 Version 0.7.1

Released on Aug. 22, 2015

- Fix inline html when there is no content per #71.

9.10 Version 0.7

Released on Jul. 18, 2015

- Fix the breaking change in version 0.6 with options: **parse_inline_html** and **parse_block_html**
- Breaking change: remove **parse_html** option for explicit
- Change option **escape** default value to `True` for security reason

9.11 Version 0.6

Released on Jun. 17, 2015

- Breaking change on inline HTML, text in inline HTML will not be parsed per #38.
- Replace **tag** renderer with **inline_html** for breaking change on inline HTML
- Double emphasis, emphasis, code, and strikethrough can contain one linebreak per #48.
- Match autolinks that do not have / in their URI via #53.
- A work around on link that contains `)` per #46.
- Add `<font>` tag for inline tags per #55.

9.12 Version 0.5.1

Released on Mar. 10, 2015

- Fix a bug when list item is blank via ipython#7929.
- Use python-wheels to build wheels for Mac.

9.13 Version 0.5

Released on Dec. 5, 2014. This release will break things.

- For custom lexers, **features** is replaced with **rules**.
- Refactor on function names and codes.
- Add a way to output the render tree via #20.
- Fix emphasis and strikethrough regular expressions.

9.14 Version 0.4.1

Released on Oct. 12, 2014

- Add option for parse markdown in block level html.
- Fix on lheading, any number of underline = or - will work.
- Patch for setup if Cython is available but no C compiler.

9.15 Version 0.4

Released on Aug. 14, 2014

- Bugfix. Use inspect to detect renderer class.
- Move all meth:*escape* to renderer. Use renderer to escape everything.
- A little changes in code style and parameter naming.
- Don't parse text in a block html, behave like sundown.

9.16 Version 0.3.1

Released on Jul. 31, 2014

- Fix in meth:*Renderer.block_code*, no need to add `\n` in `<code>`.
- Trim whitespace of code in code span via #15.

9.17 Version 0.3

Released on Jun. 27, 2014

- Add `<hr>` in footnotes renderer
- Add hard_wrap configuration for GFM linebreaks.
- Add text renderer, via #9.
- Define features for lexers available via #11.

9.18 Version 0.2

Released on Mar. 12, 2014

- Use tuple instead of list for efficient
- Add `line_match` and `line_started` property on InlineLexer, via #4

9.19 Version 0.1

First preview release.

Python Module Index

m

mistune, 17

Index

A

autolink() (mistune.Renderer method), 17

B

block_code() (mistune.Renderer method), 17
block_html() (mistune.Renderer method), 17
block_quote() (mistune.Renderer method), 17

C

codespan() (mistune.Renderer method), 17

D

double_emphasis() (mistune.Renderer method), 17

E

emphasis() (mistune.Renderer method), 18
escape() (in module mistune), 20
escape() (mistune.Renderer method), 18

F

footnote_item() (mistune.Renderer method), 18
footnote_ref() (mistune.Renderer method), 18
footnotes() (mistune.Renderer method), 18

H

header() (mistune.Renderer method), 18
hrule() (mistune.Renderer method), 18

I

image() (mistune.Renderer method), 18
inline_html() (mistune.Renderer method), 18

L

linebreak() (mistune.Renderer method), 18
link() (mistune.Renderer method), 18
list() (mistune.Renderer method), 19
list_item() (mistune.Renderer method), 19

M

Markdown (class in mistune), 20
markdown() (in module mistune), 20
mistune (module), 17

N

newline() (mistune.Renderer method), 19

P

paragraph() (mistune.Renderer method), 19
placeholder() (mistune.Renderer method), 19

R

render() (mistune.Markdown method), 20
Renderer (class in mistune), 17

S

strikethrough() (mistune.Renderer method), 19

T

table() (mistune.Renderer method), 19
table_cell() (mistune.Renderer method), 19
table_row() (mistune.Renderer method), 19
text() (mistune.Renderer method), 20